Beauty in the Crowd:

Oggero, S.

2013

6 VERTEXING WITH THE PILE-UP SYSTEM

The Pile-Up detector has been designed to reduce the fraction of accepted events with more than one interaction in the same bunch crossing. Such events are more difficult to analyse offline and should be filtered out of the data sample at the Level-zero (L0) trigger by a veto. The reason to detect multiple interactions at the earliest trigger level was to free-up bandwidth for other L0-trigger lines. For this reason, detection of multiple interactions takes place at the hardware level: the Pile-Up detector provides information to the L0-trigger system at the 40 MHz bunch crossing frequency.

The information provided to the L0 Decision Unit [68] mainly consists of the number of reconstructed primary vertex candidates and their position along the beam line (z-coordinate). In this chapter we describe the Pile-Up Veto algorithm and its optimisation. We also present a comparison of the results obtained from the Pile-Up system to those extracted from offline reconstructed events.

6.1 Pile-Up vertexing algorithm

The vertexing algorithm, implemented in FPGAs[1] on the Pile-Up Processing Boards (VEPROBs, see description in Sec. 4.3), is based on a geometrical relation valid for tracks originating from a vertex. The two PU sensors record the radial position of hits from traversing tracks: if A and B are hits on the two detector planes associated to the same track (see Fig. 51), R_A and R_B are the corresponding hit r-coordinates.
If Z_{PV} is the (unknown) z-coordinate of the primary vertex, or track origin, we can write the following relation:

$$\frac{R_B}{R_A} = \frac{Z_B - Z_{PV}}{Z_A - Z_{PV}} = k \; . \tag{86}$$

In this case, Z_A and Z_B are the positions of the two sensor planes along the beam line, and the ratio k is unique for a certain z-position.

At each bunch crossing, the algorithm combines all the readout hits (R_A, R_B) in a coincidence matrix according to Eq. 86. Figure 51 shows a sketch of the correlation plot of R_A versus R_B. From each combination of hits, indicated by a diagonal line in the plot, the corresponding vertex z-position is obtained and a new entry is added to an appropriately binned histogram, hereafter referred to as "vertex histogram". The content of each bin corresponds to the number of tracks originating from that z-position, or the vertex track multiplicity; therefore a "large" number of entries indicates the presence of a primary vertex. The vertex histogram is constrained to a window of -15 cm$<z<$15 cm, where $z = 0$ is the nominal LHC interaction point.

All algorithm steps are illustrated in a flow chart in Fig. 52. Once the histogram is filled, a parallel peak search is performed. The bin with the highest number of entries is identified as a vertex candidate in case its content is above a certain (tunable) threshold; in this case both the bin position and the height of the peak are stored in memory. If more than one bin is found with the same number of entries, the position and height of a maximum of two peaks are stored.

1 FPGAs are Field Programmable Gate Arrays.

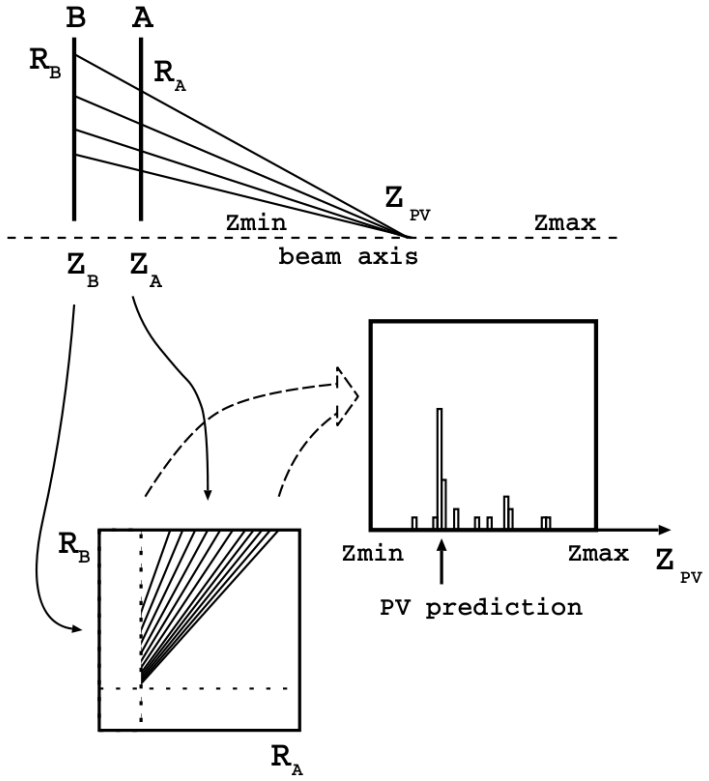


Figure 51 ……
Scheme showing the basic principle of the PU vertex algorithm implemented on the VEPROBs. The r-coordinates of the readout hits are combined in a coincidence matrix of diagonals. From each combination of hits, the corresponding vertex z_{PV}-position is obtained and a new entry is added to an appropriately binned histogram. Its highest peak is labeled as primary vertex PV.

The histogram has 128 bins with increasing bin widths towards positive z. The number of bins is chosen to constrain the time of the search algorithm. In this way, the full PU vertex algorithm takes about 33 clock cycles of 25 ns each, plus 2 for the synchronisation of the input links on the board, fitting well within the Level-0 latency requirement of 4 μs. The bin size varies from 1 to 5 mm as listed in Tab. 7. The bin widths are optimised to flatten the combinatorial background, in agreement with the PU resolution.

Table 7 ……
Pile-Up vertex histogram: bin size variation.

bin range	bin width	z range
$0 \leq$bin< 50	1 mm	$-150 \leq z < -100$ mm
$50 \leq$bin< 75	2 mm	$-100 \leq z < -50$ mm
$75 \leq$bin< 105	3 mm	$-50 \leq z < +40$ mm
$105 \leq$bin< 128	5 mm	$+40 \leq z \leq +150$ mm

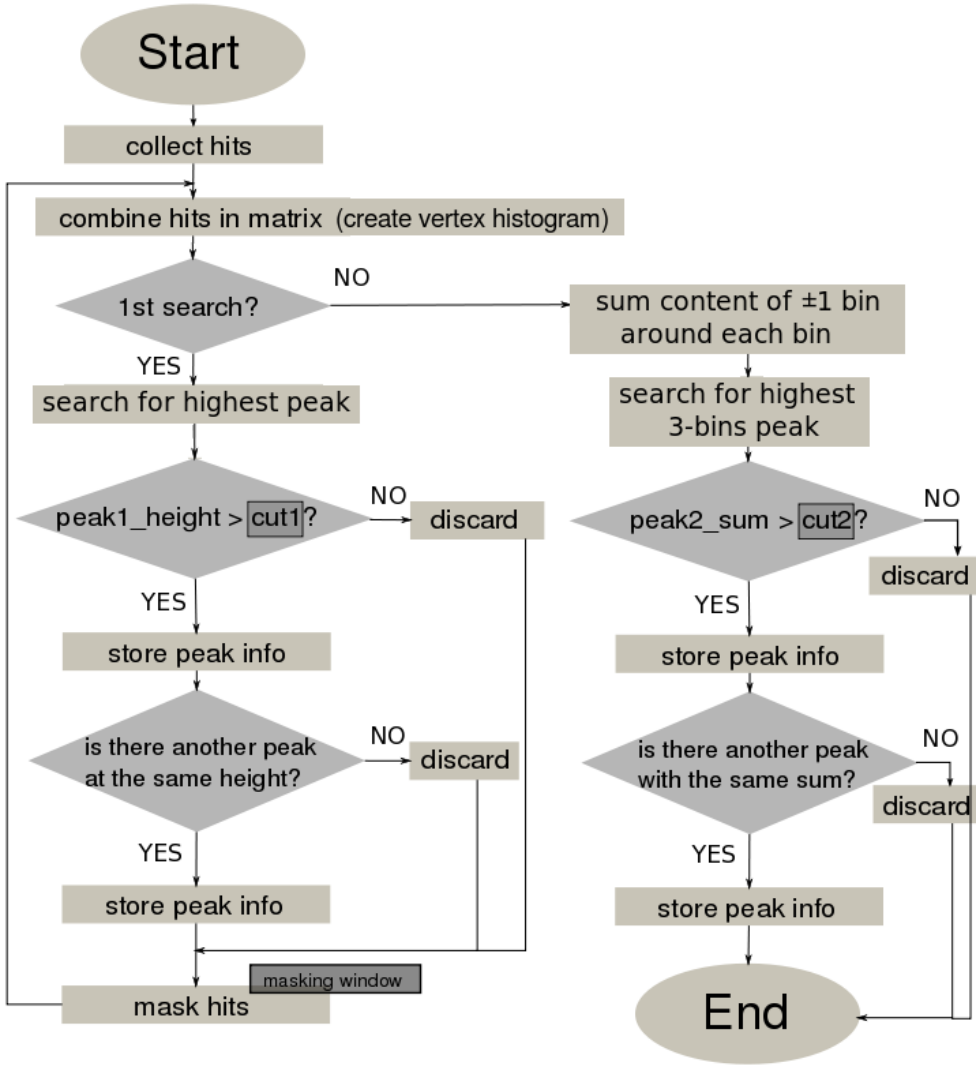


Figure 52 ……
Flow chart describing the PU algorithm as it is implemented in the hardware. The boxes in dark grey emphasise the parameters that can be set.

Figure 53 shows an example vertex histogram, from a collision event of 2010 data, produced by emulating the hardware algorithm in the LHCb software framework and running the emulator on raw data. For more information on the algorithm emulator, see Sec. 6.2.

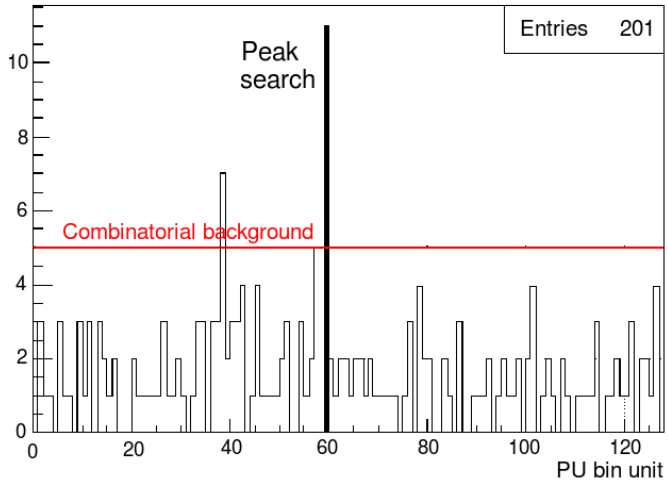


Figure 53 Example vertex histogram obtained from the combinations of PU hits of a collision event of 2010 data. The histogram is produced by emulating the hardware algorithm in the LHCb software and running the emulator on raw data. The highest bin, indicated in black, has been found during the first peak search. The second peak at bin 38 might be found during the second peak search (see explanation in the text), while the other bins form the combinatorial background.

To disentangle multiple vertices in the same event, a second peak search is performed on the FPGA, as illustrated in the right half of the flow chart of Fig. 52. Two interactions can produce a very different number of tracks and the height of the peak related to the second interaction can be lower than the overall level of combinatorics, such that it becomes indistinguishable from the background. To cope with this problem, the combinations which contribute to the first peak are identified and their hits on both planes are masked. The masking reduces the combinatorial background and improves the signal over background ratio.

The procedure is then iterated: the remaining hits are again combined in a coincidence matrix, a histogram is filled and a second peak search performed. This time the peak is identified as the bin with the highest sum of entries in a window of ± 1 bin around its position (hereafter referred to as "peak sum"). The sum is required to be above a certain threshold in order to associate the peak to a vertex candidate and store its information (position, number of entries) in memory. Figure 54 shows the vertex histogram obtained for a collision event in 2011, before (black) and after (white) masking the hits from the first peak. The second peak, with the maximum number of entries in a 3-bins wide window, is now clearly visible.

6.1.1 Tunable parameters and output bits

The current implementation of the algorithm allows to apply two thresholds before storing the peaks' information into memory. These thresholds are: the minimum number of entries

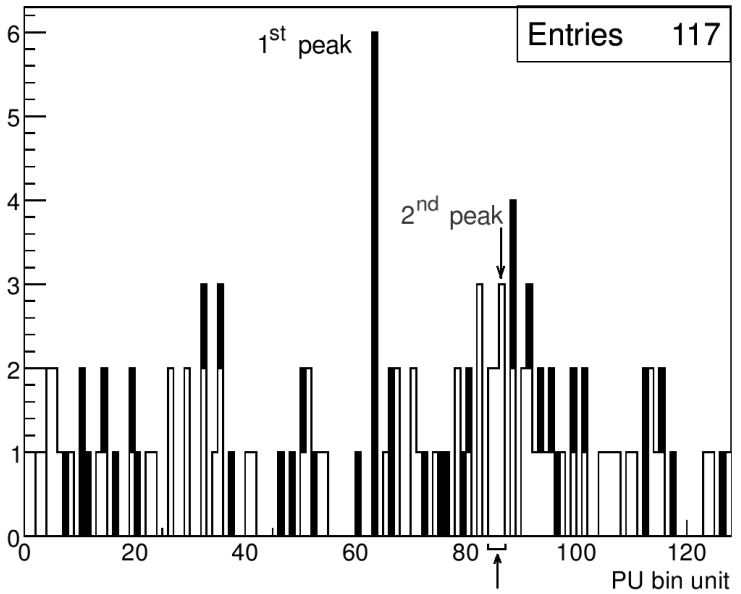


Figure 54 Vertex histogram obtained from the combinations of PU hits from a collision event of 2011 data. The histogram filled in black (white) is obtained before (after) the "peak-masking" phase. The first search finds the highest peak at bin 64. The second search finds a second peak at bin 85, that is the peak with the maximum number of entries in a 3-bins wide window.

in the bin corresponding to the first peak found and the minimum number of entries in a 3-bins wide window around the bin position of the second peak found. If none of the peaks found during the first search is above the chosen threshold, no candidate is stored in memory, but a second peak search still takes place. The thresholds are two of the parameters which allow to tune the algorithm without modifying the firmware in the FPGAs.

If we allow for modifications to the firmware, another parameter can be introduced to improve the tuning. This is the width, in terms of number of bins, of the masking window applied after the first peak is found. It allows to mask the hits from the combinations contributing not only to the central bin of the peak, but also to the neighbouring $\pm$ 1, 2, 3, 4 bins, according to the window chosen. In fact, the peaks are typically smeared and removing only the central bin would result in a second peak often found adjacent to the first one.

At the end of the two peak searches, the results are sent to the L0 Decision Unit. The Pile-Up algorithm can identify events with more than two vertices, but only the information of the first two candidates are stored. We list in Tab. 8 the L0DU words carrying information from the PU algorithm, according to the labelling assigned in the LHCb software.

Not only the results of the vertex searches are encoded into L0DU words, but also the multiplicity counter; this is chosen to be the sum of hits collected on the plane closest to the VELO, i.e. PU sensors 130 and 131. The multiplicity is currently used in several L0-trigger lines, as explained in Ch. 7.

Moreover, via the `MoreInfo` field, the total number of vertex candidates found is encoded in the L0DU bank. This is done in a way that also provides a description of how the search

Table 8 List of L0DU words carrying information obtained from the Pile-Up vertexing algorithm.

word name	meaning
`PUPeak1(Pos)`	position of the first peak, in bin units
`PUPeak1(Cont)`	content of the first peak (peak1 height)
`PUPeak2(Pos)`	position of the second peak, in bin units
`PUPeak2(Cont)`	content of the second peak (peak2 sum)
`PU(MoreInfo)`	number of vertices (0, 1, 2 or more than 2)
`PUHits(Mult)`	multiplicity counter

proceeded, and hence how reliable the candidates are. Table 9 explains how the algorithm assigns a value to the `MoreInfo` word.

Table 9 Meaning of the PU `MoreInfo` field (values from 0 to 15) in the L0DU data bank: it encodes the information on the number of vertex candidates N found by the Pile-Up algorithm. The columns refer to the number of vertices found above threshold at the 1^{st} peak search, while the rows refer to the number of vertices found above threshold at the 2^{nd} peak search.

		1^{st} search			
	N	0	1	2	>2
2^{nd} search	0	0	4	8	12
	1	1	5	9	13
	2	2	6	10	14
	>2	3	7	11	15

6.2 The algorithm emulator

In order to test and tune the online vertexing algorithm, we implemented a software package in the LHCb framework[2], emulating the FPGA algorithm offline. The emulator is organised in two parts, the first part generates a `L0PU` RawData bank (see Sec. 4.4.3) from Monte-Carlo hits, the second one emulates the output of the vertexing algorithm and can either accept (measured) RawData banks or MC hits transformed to RawBanks. The following corresponding `C++` classes are defined:

`PuVetoFillRawBuffer` **class:** it builds the hit pattern on the PU sensors, starting from MonteCarlo events. The strip signals are extracted event by event and mapped onto the sensor strip geometry. The signals are required to exceed a fixed threshold of 7000 electrons per single channel and are subsequently ORed by 4 strips to construct the digital hit pattern. The information is then encoded into 32-bit words and written in a `L0PU` RawBank in a format identical to that generated by the hardware.

2 The code is placed in the `Lbcom` components project, under the package named `L0/PuVeto`.

`PuVetoAlg` **class:** it emulates the vertexing algorithm itself, starting from the PU hit pattern. The class first runs the raw bank decoder to get the signal organised in clusters, then combines the hits in a coincidence matrix and fills a histogram in the same way as the hardware does. The peak searching and masking of the hits are also emulated. The output of the algorithm, in terms of peak position and peak height or sum, is encoded into the `L0DU` raw bank. The hit multiplicity is calculated and encoded as well, since the purpose of the class is to create a bit-perfect emulation of the algorithm that runs on the trigger hardware.

We use the `PuVetoAlg` class in sequence after `PuVetoFillRawBuffer`, to simulate the chain from MC data, as well as directly on real raw data. In this second case, we use it to compare the output of the software emulator with the hardware output and test the bit perfectness of the emulator itself. This study was performed on a sample of 2010 data and showed that the algorithm is indeed bit-perfect on the hit multiplicity measurement. The matching of the vertex candidates was at first found to be not fully bit perfect. The information encoded in the L0DU bank showed a mismatch in about 1‰ of the cases. The scatter plot in Fig. 55 (a) shows on the $x-$axis the position of the PU vertex, as it is evaluated by the software emulator, and on the $y-$axis the same value as it is stored in the L0DU Raw Bank. About 0.14% of the entries do not fall on the diagonal; the difference is also shown on a 1-dimensional plot in Fig. 55 (b).

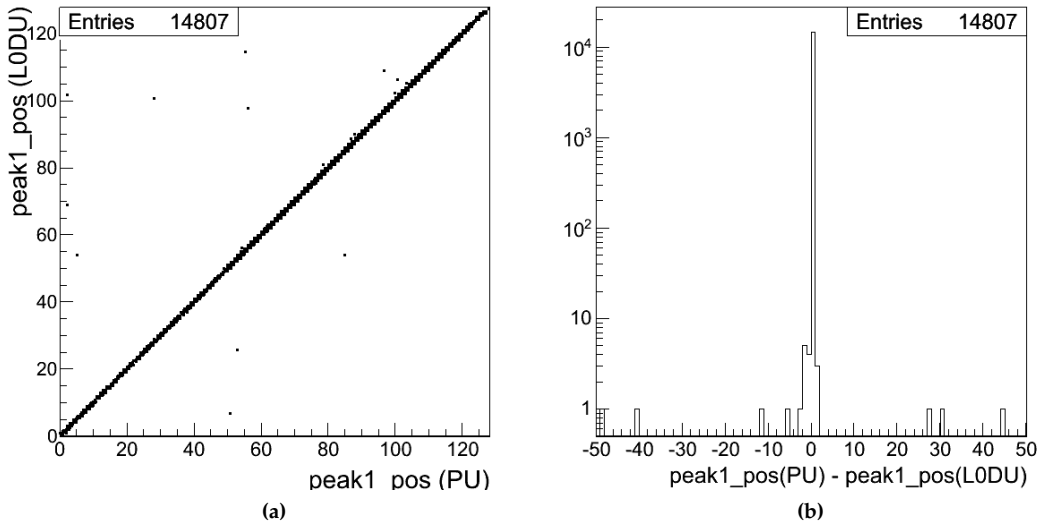


Figure 55
The histogram in (a) shows on the $x-$axis the position of the PU vertex found at the first search, as it is evaluated by the software emulator, and on the $y-$axis the same value as it is stored in the L0DU Raw Bank. The same information is shown in (b), where on the $x-$axis the difference between PU vertex position and L0DU vertex position is shown in bin units.

By performing a hardware simulation of the algorithm on the FPGA board, the emulator and the hardware algorithm were shown to fully match. After further studies, it was found that the observed mismatches were instead caused by transmission of double-sized raw data banks to the L0DU board. A correction to the firmware guaranteed the transmission of

correctly sized banks and cured the mismatch. The 2012 PU firmware gives a 100% match between hardware output and L0DU information.

6.3 Tuning of the algorithm for beam-beam collisions

In this section we present a study on the optimisation of the Pile-Up algorithm, to maximise the vertexing performance in comparison to that of the LHCb offline reconstruction. The work focuses on a direct event-by-event comparison of the output of the PU algorithm emulator, described in the previous section, to the output of the offline VELO vertexing algorithm applied on the same minimum bias data sample from 2010. About 70k events are used for this analysis. The vertices reconstructed by the VELO are considered as a "reference point", as justified by the high resolution and track finding efficiency in the VELO [99].

In Sec. 6.3.1 we present a first PU-VELO performance comparison, before algorithm optimisation, followed by a description of the optimisation method. The optimised parameters and obtained results are then listed in Sec. 6.3.2.

6.3.1 First VELO-PU comparison and optimisation choice

The main variable used to optimise the PU algorithm is the number of vertices found by the Pile-Up detector. For this comparison, the offline VELO vertices are required to have a z-position between -15.5 cm and +15.5 cm, to include only vertices inside the PU acceptance window. To select primary vertices, we also require the number of tracks per VELO vertex to be higher than 10.

Figure 56 shows a comparison between the number of VELO offline vertices and the number of vertex candidates obtained with the Pile-Up algorithm with its initial settings, i.e. before optimisation. Often, the PU detects a higher number of vertices than what is reconstructed offline. This is due to the fact that the parameters initially configured in the hardware algorithm correspond to relatively loose requirements: a masking window of ± 1 bin and a minimum threshold of 2 on the peak1 height and peak2 sum. Note that to produce the histogram, a slightly extended implementation of the algorithm emulator is used; this will be introduced in Sec. 6.3.2.

To evaluate the performance of the PU algorithm, its specific use in the LHCb trigger should be taken into account. A different use of the algorithm will lead to a different optimisation choice. In this study, the optimisation is performed in view of using the algorithm as originally intended, i.e. to veto collisions with more than one vertex. Assuming the offline VELO vertexing to be correct, the algorithm is considered efficient when accepting events with 1 VELO vertex and rejecting events with more than 1 VELO vertex and inefficient in all other cases. This is illustrated with different colour codes in Fig. 56, blue for efficient cases and red for inefficient ones. If the VELO observes 1 or 0 vertices, while the PU algorithm identifies more than 1 vertex candidate, than the Pile-Up would wrongly reject a potentially interesting physics event. On the other hand, events with at least 2 VELO vertices but less than 2 PU vertex candidates would be wrongly accepted, filling a part of the L0-trigger bandwidth with potentially uninteresting events.

With the initial settings that lead to Fig. 56, 24.5% events are correctly accepted and 34% are correctly rejected, while 36.1% events are wrongly rejected and 5.4% are wrongly accepted by the PU algorithm. This corresponds to a fraction (hereafter named "total efficiency") of correctly assigned events, in the above definition, of 58.5%.

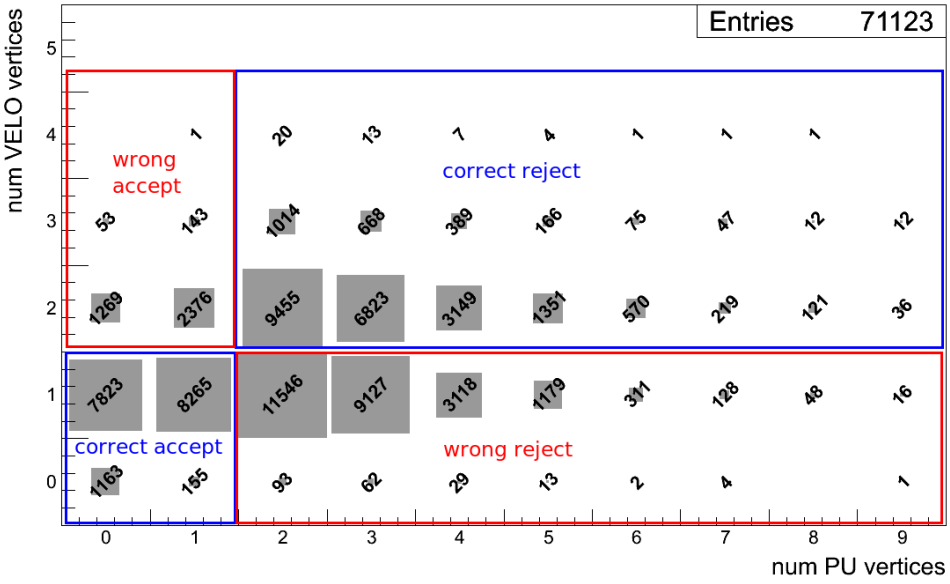


Figure 56 The histogram is produced with a minimum bias data sample from 2010; it shows on the x-axis the number of vertex candidates found by the PU algorithm and on the y-axis the number of VELO vertices reconstructed offline for the same events. The PU algorithm is emulated with a masking window of ± 1 bin and cuts of peak1 height$>$2 and peak2 sum$>$2, corresponding to the initial set of configured in the hardware.

The optimisation aims at an increase of such efficiency, within the available bandwidth dedicated to a PU-based trigger output. This bandwidth will be filled by events with 1 visible[3] pp interaction efficiently triggered by the PU, and events with more than 1 visible pp interaction that fail to be rejected (i.e. inefficiently rejected) by the PU. Assuming that the number n of visible pp interactions per beam crossing follows a Poisson distribution with parameter μ

$$P(n;\mu) = \frac{\mu^n e^{-\mu}}{n!} \ , \tag{87}$$

the PU output rate can be calculated as the sum of two contributions:

- $R(1)$, PU rate of (correctly) triggered events with 1 visible pp interaction
- $R(>1)$, PU rate of (wrongly) triggered events with more than 1 visible pp interaction

Therefore:

$$R(1) = N_{bunches} \cdot f \cdot P(1) \cdot \epsilon_1^{PU} \ , \tag{88}$$

$$R(>1) = N_{bunches} \cdot f \cdot P(>1) \cdot (1 - \epsilon_{>1}^{PU}) \ , \tag{89}$$

where ϵ_1^{PU} and $\epsilon_{>1}^{PU}$ are the efficiencies to correctly keep or correctly reject events, respectively. Moreover, $P(>1)$ = 1-$P(0)$-$P(1)$ and f = 11245 Hz is the beam revolution frequency.

A factor to consider is the time required by the online farm to process events, once those are selected by the PU algorithm. The processing time will increase with the number of

3 The word "visible" is used to label pp interactions leaving "some" decay products in the detector, therefore the number of visible pp interactions depends on the definition of "visible" in LHCb.

vertices contained in the event. Therefore, we aim at filling the output rate of PU-triggered events with the highest possible number of single-vertex events and the lowest possible number of multiple-vertices ones.

If w_1, $w_{>1}$ are weights assigned to the two categories, proportional to the corresponding processing time, we can write

$$R_{PUout} = w_1 \cdot R(1) + w_{>1} \cdot R(>1) \ . \tag{90}$$

For the study presented in this dissertation, we assume that the event processing time is not a limiting factor, hence we choose $w_1 = w_{>1} = 1$.

Additionally, for the algorithm optimisation, we assume that the beam crossings with zero interactions in LHCb are already rejected by other L0-triggers (e.g. a trigger line requiring a minimum energy deposit in the calorimeters). Therefore, all events that do not contain any VELO vertex are discarded from the analysed sample.

The optimisation depends on the one hand on the beam crossing rate ($N_{bunches} \cdot f$) and the average pile-up μ, and on the other hand on the available bandwidth. Here we optimise on a data sample with $N_{bunches} = 16$. Concerning μ, a pp interaction is visible in LHCb if it contains at least 2 `VeloRZ` tracks[4]. Hence, the value of μ is extracted by counting how many trigger unbiased events[5] in the analysed dataset are empty (i.e. they have less than 2 `VeloRZ` tracks). For this particular run, μ is computed to be 0.989 and leads to $P(1) = 0.365$ and $P(>1) = 0.264$.

Therefore, to evaluate the PU efficiencies entering equation 90, we first split the analysed events in two categories: N_1 events with strictly one VELO vertex and N_2' events with more than one. We define

- ϵ_1, the PU (relative) efficiency of triggering events with 1 VELO vertex, computed as number of events with 1 VELO vertex and 0 or 1 PU vertex divided by N_1;
- $\epsilon_{>1}$, the PU (relative) efficiency of rejecting events with more than 1 VELO vertex, computed as number of events with at least 2 VELO vertices and at least 2 PU vertex candidates divided by N_2.

The goal of the optimisation described in the next paragraph is to find an optimal balance between a high $P(1) \cdot \epsilon_1$ and a low $P(>1) \cdot \epsilon_{>1}$, while keeping the total output rate below the available bandwidth R^{max}_{PUout}.

6.3.2 Optimisation of the parameters

The optimisation study is performed with the emulator sketched in Fig. 57. It allows to store the exact information on the number of vertex candidates found at each search, instead of storing it in terms of 0, 1, 2, or more than 2 vertices in the `MoreInfo` field, as explained in Sec. 6.1.1.

Initially, only the masking window width is varied, while the other parameters are kept fixed. This is done to understand the dependency of the algorithm performance on the masking window. Figure 58 (a) shows in black the efficiency values obtained for ϵ_1, with masking windows of ±1, ±2, ±3 and ±4 bins, respectively, for the two initial thresholds on the peaks (peak1 height$>$2 and peak2 sum$>$2). As a comparison, the same histogram is produced applying stronger peak cuts (peak1 height$>$3 and peak2 sum$>$3) and is shown in red. In Fig. 58 (b) equivalent distributions are shown for the obtained values of $\epsilon_{>1}$. The study shows that a wider masking window gives a lower efficiency to find a second peak,

4 For a definition of `VeloRZ` tracks, see paragraph 3.3.3.

5 We use the L0-NoBias trigger line.

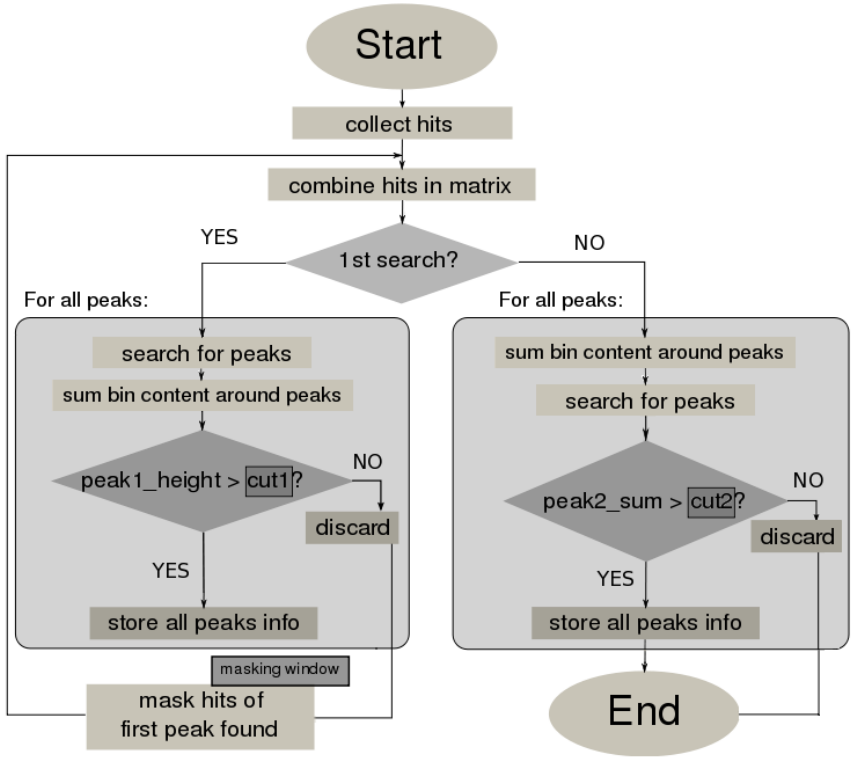
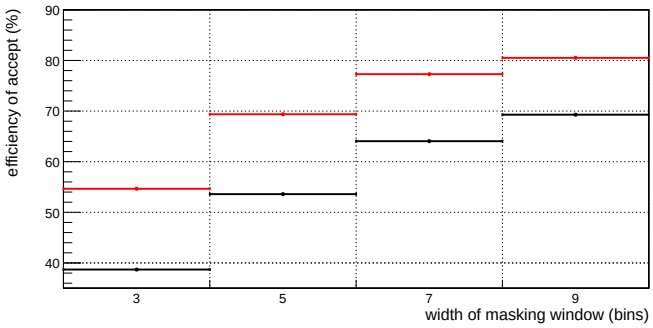
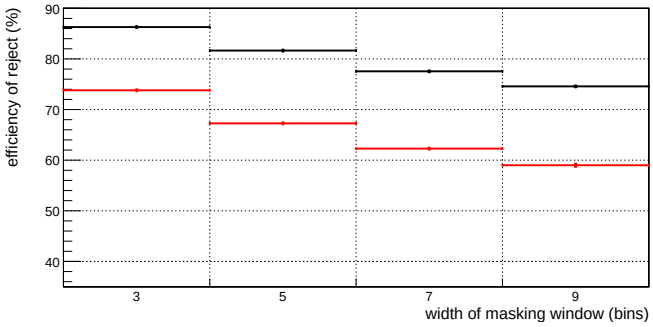


Figure 57 ……
Flow chart describing the software emulation for an improved version of the PU algorithm, allowing to store the exact information on the number of vertex candidates found at each search.



(a) Efficiency to accept ϵ_1^{PU}



(b) Efficiency to reject $\epsilon_{>1}^{PU}$

Figure 58
The histogram in (a) shows the PU efficiency to accept events with only 1 offline vertex for masking windows of ± 1, ± 2, ± 3 and ± 4 bins (equivalent to widths of 3, 5, 7 and 9 bins respectively), when applying a cut on peak height/sum$>$2 in black, or a peak height/sum$>$3 in red. The histogram in (b) shows the same distribution with analogous colour code, but for the PU efficiency of rejecting events with more than 1 offline vertex.

and therefore a lower probability to reject the event and a higher probability to accept it. As shown in the histograms, a wider masking window leads to an increase of the efficiency to accept events with one VELO vertex and a decrease of the efficiency to reject events with more than one VELO vertex. Moreover, while the efficiency values obtained are strongly dependent on the peak cuts, the increasing or decreasing trend due to the masking window is not dependent upon them.

In order to choose the optimal combination of masking window and peak thresholds, it is necessary to balance the two efficiencies accounting for the run conditions, that is for $P(1)$ and $P(>1)$. The values of $R(>1)$ versus $R(1)$, obtained for all possible peak thresholds ranging between 1 and 8 and for the four considered masking windows, are shown in Fig. 59. The colour code is used to differentiate between different masking window widths. The choice of the best parameter combination, corresponding to a certain point on the histogram, depends on the maximum allowed output rate. For the case where the maximum PU output rate R_{PUout}^{max} = 70 kHz and assuming $w_1 = w_{>1} = 1$, we can draw on the same figure the corresponding line obtained from Eq. 90 as

$$R(>1) = -R(1) + R_{PUout}^{max} \ . \tag{91}$$

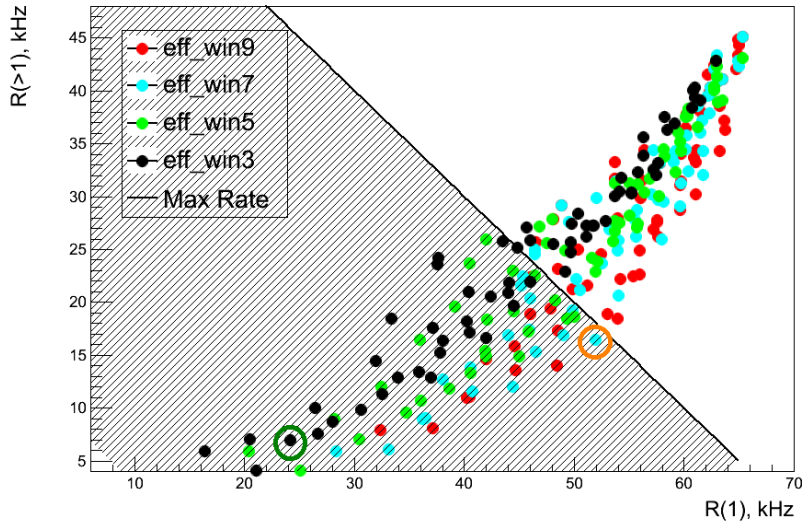


Figure 59 .....
Values obtained for $R(>1)$ versus $R(1)$, for all possible thresholds on peak1 height and peak2 sum (ranging between 1 and 8) and for the four considered masking windows of ± 1, ± 2, ± 3 and ± 4 bins, in black, green, light blue and red, respectively.

All points below that line, in the dashed area, remain within the allowed bandwidth and are therefore acceptable. Among them, we favour the combination of parameters listed in Tab. 10 giving the highest value of $R(1)$ and the lowest value of $R(>1)$ at the same time. In fact, this combination -corresponding to the point circled in orange- gives a PU efficiency of correctly accept events with 1 visible pp interaction of $\epsilon_1^{PU} = 77.2\%$ and a PU efficiency of correctly rejecting events with 2 or more visible pp interactions of $\epsilon_{>1}^{PU} = 62.3\%$. Note that

Table 10 .....
Optimal PU parameters used for vertexing studies.

masking window width = 7 (± 3 bins)
`peak1 height` > 3
`peak2 sum` > 3

the initial combination of settings corresponds to the point circled in green. It falls in the allowed bandwidth, but gives a low efficiency of correctly accept events with 1 visible pp interaction[6].

As a reference, the values of $R(>1)$ versus $R(1)$ obtained for all possible peak thresholds ranging between 1 and 8 and for a masking window of ± 3 bins are shown in Fig. 60. Here the colour code is used to differentiate between different thresholds on peak2 sum, while

6 Viceversa, the initial combination would correspond to the optimised case at highest $R(1)$ and lowest $R(>1)$ for a much smaller bandwidth of about 30 kHz.

each point on the same curve corresponds to a different cut on peak1 height, at fixed peak2 sum threshold. The chosen parameter combination is again highlighted with an orange circle, while the maximum allowed output rate defines the position of the black line on the plot.

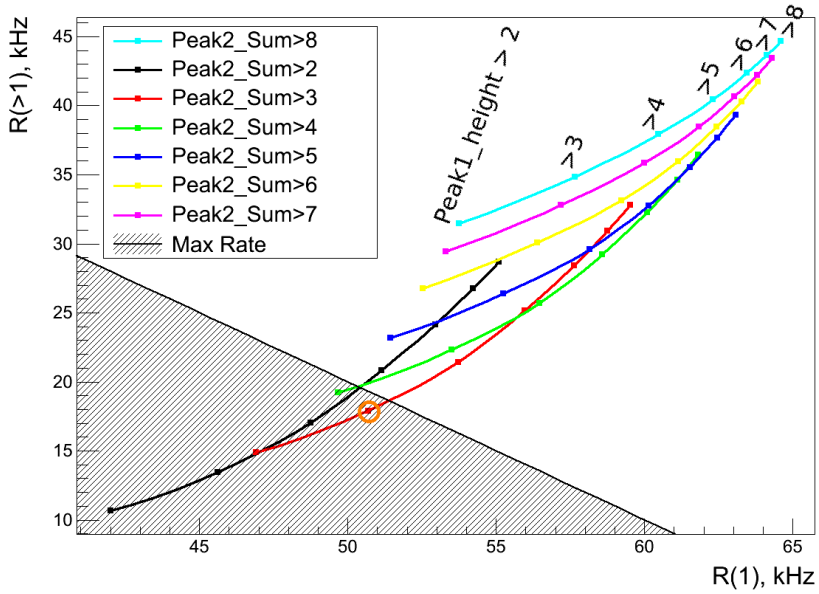


Figure 60 Values obtained for $R(>1)$ versus $R(1)$, for all possible thresholds on peak1 height (varying from 2 -left- to 8 -right- on each curve, as labelled on the blue one) and peak2 sum (ranging on the colour code between 2 and 8, as indicated in the legend), for a masking windows of ± 3 bins.

Finally, Fig. 61 shows the comparison between the number of vertex candidates obtained with the Pile-Up algorithm and the number of offline vertices, for the optimised algorithm. We compute: 47.1% of the events correctly accepted, 24.5% correctly rejected, 13.5% events wrongly rejected and 14.9% wrongly accepted, corresponding to an increase of the total efficiency –fraction of correctly assigned events– from 58.5% for the initial settings (see Sec. 6.3.1) to 71.6%. Nevertheless, the optimisation also leads to a reduction of the number of correctly rejected events and an increase of the number of wrongly accepted ones, due to the allowed bandwidth. In other words, the algorithm is more efficient on events with one VELO vertex than on events with more than one[7].

6.4 Position of PU vertices

So far, the study focused on the number of PU vertices obtained, as this is the information available in the L0DU that can be used to veto events. It is also interesting to study the

7 This is not surprising, since the main focus was on maximising ϵ_1 within the rate limit, via $R(1)$, while $\epsilon_{>1}$ is not maximised.

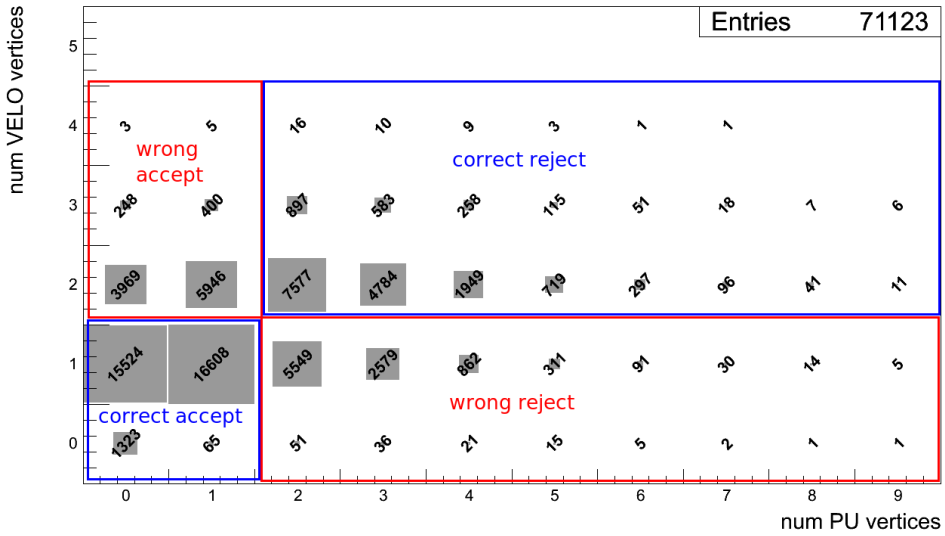


Figure 61 The histogram shows on the x-axis the number of vertex candidates found by the PU algorithm and on the y-axis the number of VELO vertices reconstructed offline for the same events. The PU algorithm is emulated with a masking window of ± 3 bins and cuts of peak1 height>3 and peak2 sum>3, corresponding to the set of parameters optimised for a maximum output rate of 70 kHz.

z-position of the PU vertex candidates obtained, in comparison to the VELO ones. This final check does not aim at improving the performance of the algorithm, but at verifying that the PU and offline vertexing match well.

From the same 2010 sample used for the optimisation study, we select events with strictly 1 VELO vertex and we implement the PU vertex algorithm according to the parameters given in Tab. 10. We also require the number of tracks associated to each offline vertex to be higher than 10. The VELO primary vertex resolution depends on the track multiplicity, as shown in Fig. 62, and it falls below 150 μm in the z-direction for n_{tracks} >10.
For each event and each VELO vertex, the distance in z to the nearest PU vertex candidate is determined. Figure 63 (a) shows the obtained distribution, fitted with a Gaussian function. The result proves that the Pile-Up is able to detect vertices with an average resolution of about 4 mm in the $z-$position. Note that misaligned PU data are shown here, as this corresponds to the information available at the L0 trigger.

Figure 64 shows the same distribution, but now obtained by splitting the sample according to the z-position of the VELO vertex. Each category corresponds to a range in z where the PU vertex histogram has bins of constant size, as shown in Tab. 7. From the σ of the fitted distributions we can evaluate the PU resolution at different z-positions of the vertex candidate. For instance, in the first range with -150 mm$<z_{PV}^{Reco}<$-100 mm, where each bin of the PU vertex histogram corresponds to 1 mm in z, we obtain a resolution of $\sim$1.2 mm; in the second range each bin of the PU vertex histogram corresponds to 2 mm in z and the vertex resolution is $\sim$2 mm, and so forth. Moreover, the chosen masking window of ± 3 bins (see optimised set in Tab. 10) corresponds to ± 3, 6, 9, and 15 mm respectively in these four ranges of different vertex z-position. Therefore, the chosen bin masking in each case

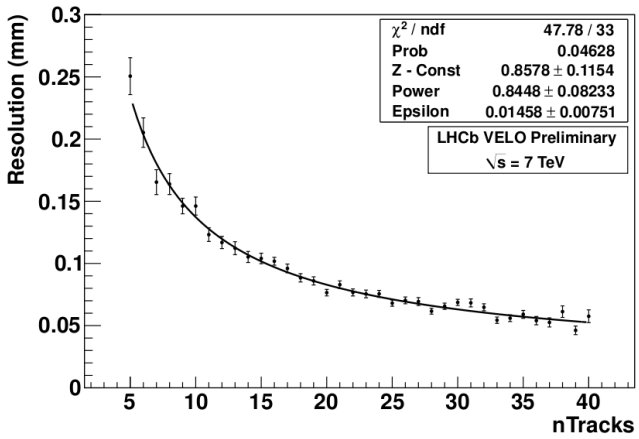


Figure 62 Primary vertex resolution in the z-direction versus number of tracks, evaluated on events with exactly one PV; figure taken from Ref. [72].

corresponds to about $\pm 3\sigma$ in terms of PU resolution, in accordance with the optimisation of the PU histogram bin width.

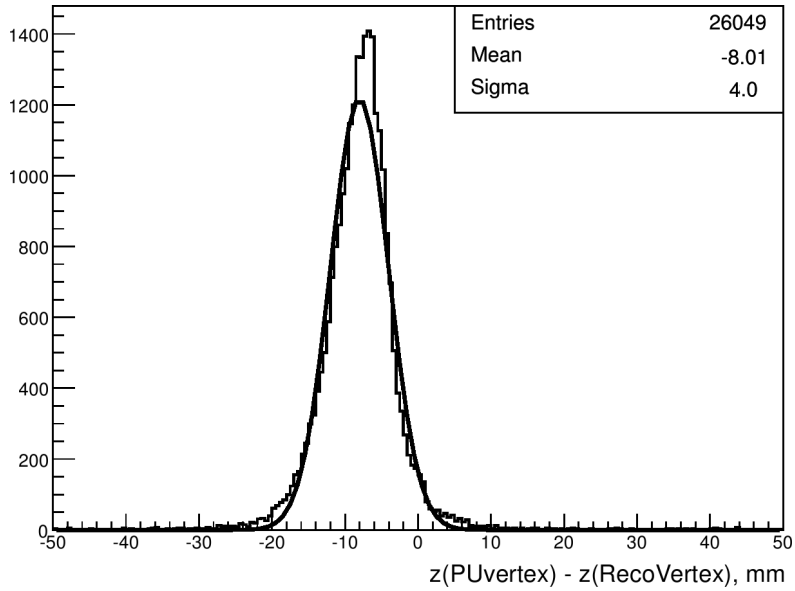
Finally, in Fig. 63 (a) we also notice a shift of several millimetres of the PU vertices towards lower values of z. This effect is mainly due to the misalignments in the detector position, and in particular to translations along the $z-$coordinate. In fact, a considerable improvement is achieved by applying the alignment corrections of the PU sensors, as described in Ch. 5. Figure 63 (b) shows the distribution of the $z-$distance between PU vertices and reconstructed ones, after the corrections to the PU sensors z-position are applied in the vertex algorithm emulator[8]. The resolution is now slightly improved ($\sim$ 3.2 mm) and the peak is closer to 0, with a mean of (-2.2 ± 0.1) mm. The same distance distribution, now obtained by splitting the sample according to the z-position of the VELO vertex, is shown in Fig. 65.

6.5 Concluding remarks

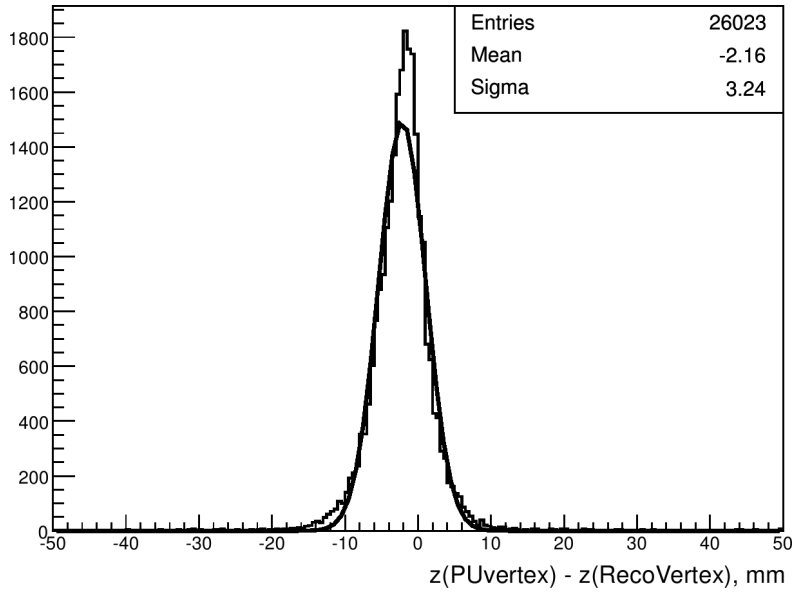
The Pile-Up detector was designed to reject events with multiple vertices at the L0-trigger level, in order to optimise the event rate with single-interaction vertices. In the mean time, experience with the LHCb data has shown that B-physics in an environment with multiple vertices is possible, such that the original veto functionality is no longer required to obtain cleanly reconstructed B-decay signals.

However, we demonstrate in this chapter that the Pile-Up algorithm is able to detect vertices in multiple interactions, with an average resolution of $\sim$3 mm along the $z-$axis (after alignment corrections). We have also shown that rejecting events with more than one interaction with the Pile-Up is feasible. For a data run with an average number of visible

8 Only z-position corrections are applied, since the vertex coincidence matrix depends, to the first order, on the position in z of the PU sensors. Note that to implement such a correction at the hardware level would signify to re-tune the hit association in the matrix.

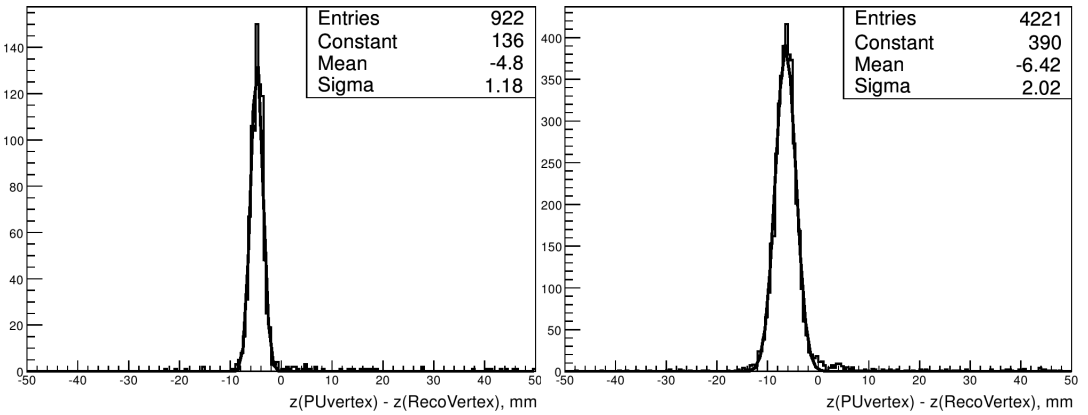


(a) Before alignment corrections.



(b) After alignment corrections.

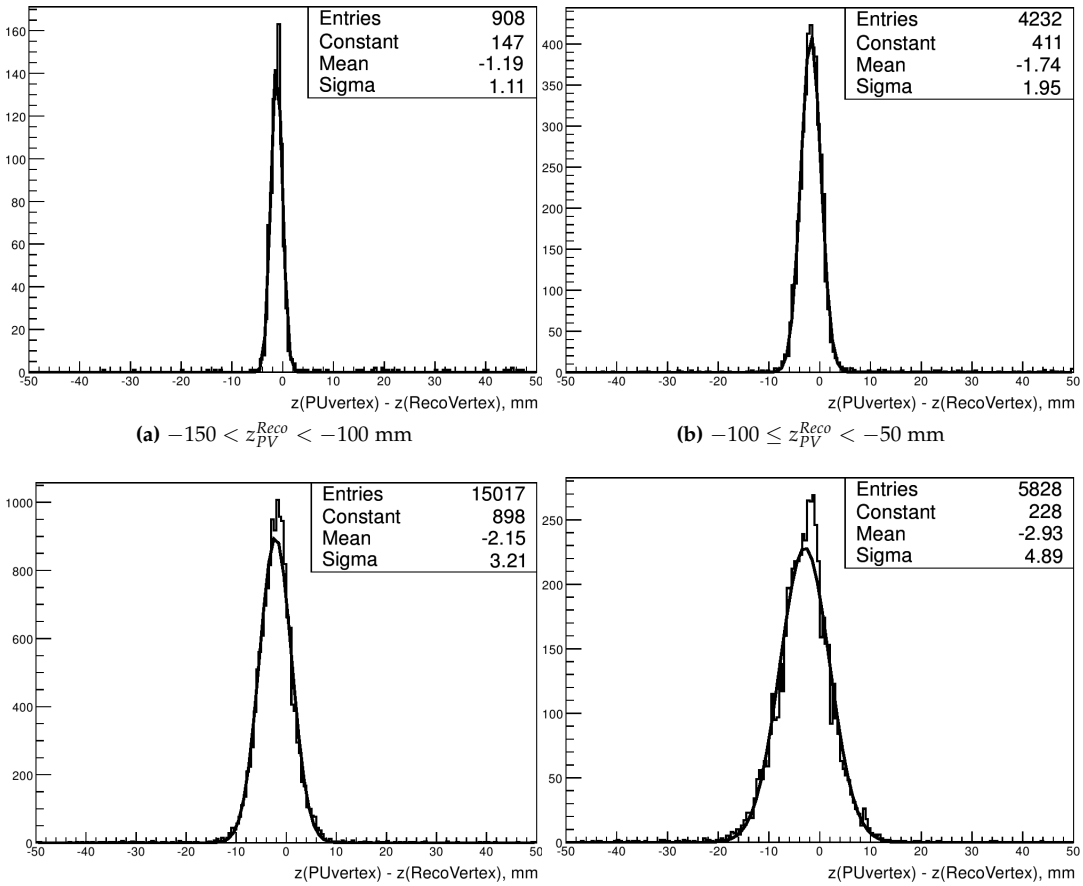
Figure 63 ……
Distance in mm between the $z-$position of the PU vertex candidate and the $z-$position of the offline vertex, for events with 1 offline vertex. The histogram in (a) is obtained before applying the misalignment corrections to the Pile-Up algorithm, while the one in (b) is obtained after.



(a) $-150 < z_{PV}^{Reco} < -100$ mm **(b)** $-100 \leq z_{PV}^{Reco} < -50$ mm

(c) $-50 \leq z_{PV}^{Reco} < 40$ mm **(d)** $40 \leq z_{PV}^{Reco} < 150$ mm

Figure 64 ……
Distance in mm between the $z-$position of the PU vertex candidate and the $z-$position of the offline vertex, for events with 1 offline vertex and for a PU algorithm optimised with the parameters in Tab. 10. The sample is split according to the z-position of the VELO vertex.



(a) $-150 < z_{PV}^{Reco} < -100$ mm **(b)** $-100 \leq z_{PV}^{Reco} < -50$ mm

(c) $-50 \leq z_{PV}^{Reco} < 40$ mm **(d)** $40 \leq z_{PV}^{Reco} < 150$ mm

Figure 65 Distance in mm between the $z-$position of the PU vertex candidate and the $z-$position of the offline vertex, for events with 1 offline vertex and for a PU algorithm optimised with the parameters in Tab. 10 and after applying the misalignment corrections to the Pile-Up algorithm. The sample is split according to the z-position of the VELO vertex.

interactions per bunch crossing $\mu \sim 1$ and assuming a total available bandwidth of 70 kHz, 47.1% of the events are correctly accepted and 24.5% are correctly rejected.

The obtained efficiency is particularly limited in rejecting events with more than one vertex. This might be partially related to the limited detector resolution in resolving peaks, as a consequence of the Pile-Up geometry. In addition, we have to account for noisy and dead channels. On PU sensor 129 for instance, 44 digital channels are masked. Together with this known dead region, the detector also developed a problem in August 2010, when sensor 128 stopped transmitting signals. The problem is due to a reset line which is permanently activated, as a result of a leakage current in one of the Beetle chips. A repair is not possible on the short term, as the Pile-Up detector is part of the VELO mechanical structure, and opening the detector is a lengthy procedure which requires breaking the LHC vacuum.

The dead sensor does not affect any of the current L0-trigger lines, since they rely on the multiplicity measurement, based only on sensors 130 and 131. For the vertexing algorithm however, it leads to a reduced performance, because it implies the use of hits only coming from half of the detector. As a consequence, the Pile-Up vertex algorithm fails to find vertex candidates more frequently, as shown in Fig. 66.

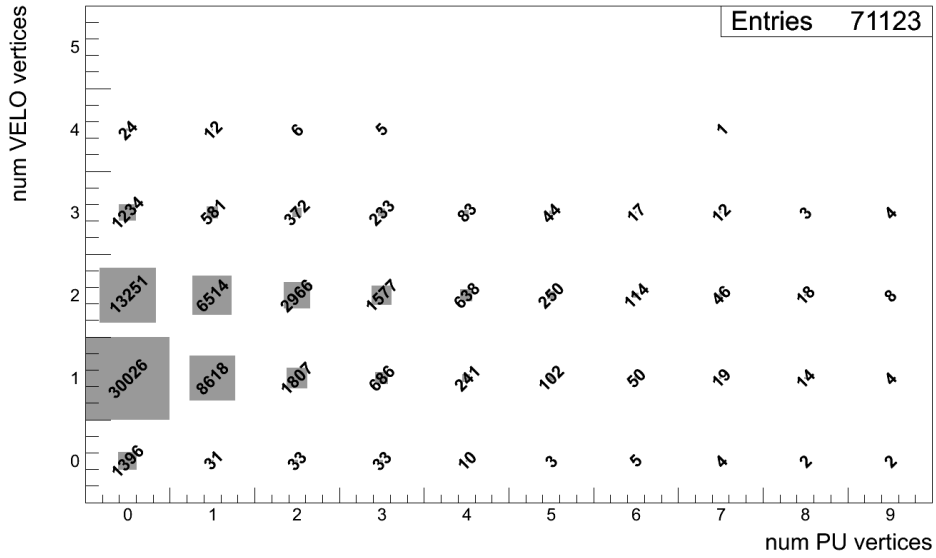


Figure 66 ……
The histogram shows on the x-axis the number of vertex candidates found by the PU algorithm and on the y-axis the number of VELO vertices reconstructed offline for the same events. The PU algorithm is emulated with a masking window of ± 3 bins and cuts of peak1 height>3 and peak2 sum>3, corresponding to the optimal set of parameters. Only the hits collected by half of the detector are used, to emulate a half-operational system.

Emulating a half operational detector with the set of optimised parameters listed in Tab. 10, we compute $\sim$56% events correctly accepted by the PU and $\sim$9% correctly rejected. The total fraction of correctly assigned events decreases from 71.6% to 65%.

Despite the hardware issues, the inclusion of the Pile-Up Veto for the coming 2014-2016 run can still be considered, especially focused on enriching the single interaction event sample. This would require not only a new optimisation study, dependent on the LHC bunch crossing scheme and on the available bandwidth, but also the opening of the detector itself and the repair of the broken sensor.

Besides, the Pile-Up detector has been used in the L0-trigger to provide a distinction between beam1-gas and beam2-gas. This will be explained in more detail in the following chapter.